Darwintrees for action recognition

Albert Clapés
University of Barcelona
Computer Vision Center
aclapes@cvc.uab.cat

Tinne Tuytelaars
KU Leuven, ESAT-PSI, imec
tinne.tuytelaars@esat.kuleuven.be

Sergio Escalera
University of Barcelona
Computer Vision Center
sergio@maia.ub.es

Abstract

We propose a novel mid-level representation for action/activity recognition on RGB videos. We model the evolution of improved dense trajectory features not only for the entire video sequence, but also on subparts of the video. Subparts are obtained using a spectral divisive clustering that yields an unordered binary tree decomposing the entire cloud of trajectories of a sequence. We then compute videodarwin on video subparts, exploiting more finegrained temporal information and reducing the sensitivity of the standard time varying mean strategy of videodarwin. After decomposition, we model the evolution of features through both frames of subparts and descending/ascending paths in tree branches. We refer to these mid-level representations as node-darwintree and branch-darwintree respectively. For the final classification, we construct a kernel representation for both mid-level and holistic videodarwin representations. Our approach achieves better performance than standard videodarwin and defines the current state-of-the-art on UCF-Sports and Highfive action recognition datasets.

1. Introduction

The task of action (or activity) recognition from image sequences is a challenging pattern recognition problem with many real-world applications, including human-computer interaction, surveillance, health-care, and video indexing, just to mention a few. Despite major recent advances in the field of computer vision, the action recognition problem still remains largely unsolved.

The extra temporal dimension added by videos in respect to images makes it harder for all kinds of predictive models. The most successful methods are those combining deep learning-based methods with pre-computed features (e.g. optical flow [25]) and/or hand-crafted methods [31, 14, 26]. The most relevant hand-crafted approach is the improved dense trajectories (iDTs) [29] framework, which consists of the extraction of short trajectories of pixels across contiguous frames followed by the computation of state-of-the-art

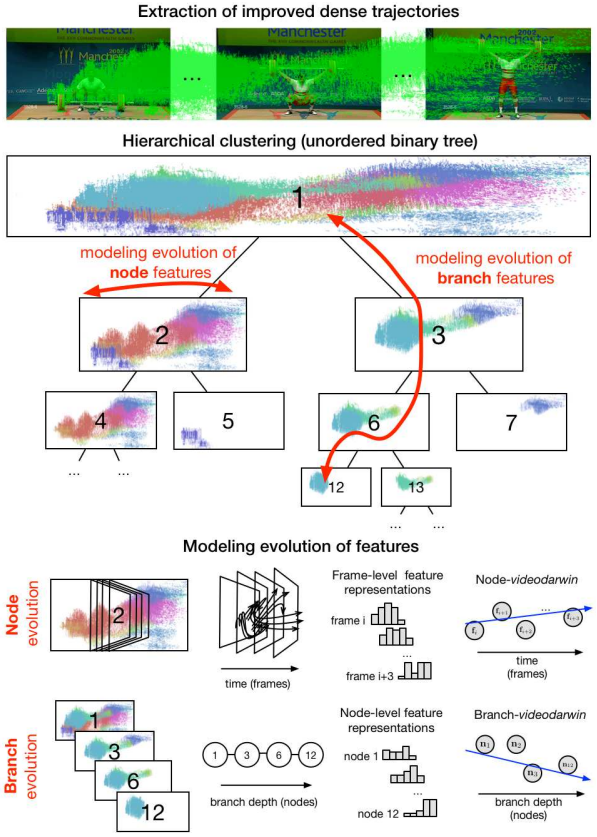

Figure 1: After the extraction of improved dense trajectories (green), we run a divisive clustering algorithm in order to obtain meaningful groupings of trajectories. Then, we perform *videodarwin* both on nodes (modeling the evolution of node frame features) and on tree branches (modeling the evolution of node global representations).

representations (HoG, HOF, and MBH) along the trajectories. Then, fisher vectors (FVs) serve as a global representation encoding all the local trajectory features observed from the video. The main drawback of FVs is however the lack of temporal information, which is not modeled explicitly.

In order to model temporal information, an alternative to

a global FV representation is to model the evolution of features within consecutive video frames using rank-pooling. Fernando et al. [8, 7] combine a per-frame FV feature representation along with an SVR/RankSVM that learns for each video the ordering of these features. Then, the parameters of the ranking machine are used as a new video representation. One of the key ingredients of this method is the preprocessing, consisting of per-frame smoothing of the features along the temporal dimension prior to ranking/regression. In practice a time-varying mean operation is used: at each time instant, per-frame features are averaged with features at previous instants. This stabilizes the signal and hence regularizes the ranking procedure. However, this same operation has the side effect that later frames see their new information vanished because the representation has already accumulated so much information. Given that, the authors of [8, 7] complemented videodarwin with the reverse operation (from the end to the beginning of the video) in order to capture the information somehow missed when performing forward videodarwin. While this makes the method more robust for relatively short videos, this solution is ill-conditioned for larger ones.

Our hypothesis is that a meaningful spatio-temporal segmentation of the video would be a simple yet effective way to obtain shorter video parts that would potentially suffer less from the smoothing problem caused by the time varying mean. Gaidon et al. [11] already proposed an unsupervised algorithm, based on spectral embedding, to build an unordered binary tree of dense trajectory clusters. They compute a bag-of-words representation in each cluster and use a tree-distance kernel for classification. They demonstrate the use of clusters yields a better result in terms of action recognition.

In this work, action patterns take the form of unordered binary trees of richly represented nodes, namely *darwintrees*. First, we cluster trajectories as in from [11]. Then, we use videodarwin to model the evolution of the features not only in the tree nodes, but also along the tree branches (see Figure 1). Using the latter, we model information of how the clusters and their parents relate as we keep dividing/grouping them. This is not only a change of paradigm in videodarwin – initially thought to model the evolution of features throughout time –, but also a way of describing variable-sized tree branches that in our experiments demonstrated to be reliable for classification providing complementary information w.r.t. the node representations. Figure 2 illustrates the different stages of the proposed framework for action recognition. We achieve state-of-the-art results in UCF Sports Actions and Highfive datasets. Moreover, differently from deep-learning methods, our method does not require high amounts of data in order to generalize.

The remainder of the paper is organized as follows: Section 2 reviews the related work; Section 3 introduces our approach; Section 4 brings details about benchmarking datasets, implementation, parametrization, results, and discussion. Finally, Section 5 concludes the paper.

2. Related work

Major advances in this pattern recognition problem that is action recognition were made with the apparition of dense trajectories [27]. The approach consists of tracking densely sampled points in a video sequence using optical flow. The pooled trajectories are then described by their relative displacement along with some trajectory-aligned descriptors, such as HOG (histogram of oriented gradients), HOF (histogram of oriented optical flow), and MBH (motion boundary histogram). Since trajectories are local descriptors, a Bag-of-Words (BoW) approach is followed for final video classification. Later, in [29], the same authors introduced *warped-flow*, which consists of computing the homography between consecutive frames in order to eliminate the optical flow caused by the camera movement and switched the global representation from BoW to fisher vectors (FV) [21].

In [11], Gaidon et al. propose an unsupervised algorithm ithat decomposes the set of trajectories into a hierarchy of trajectory clusters. They then compute a kernel based on the similarity between a pair of trees, that is, on the accumulated similarities between tree edges (parent-child node pairs). In [21], Peng et al. enrich the holistic FV representation with so-called stacked fisher vectors. Fisher vectors are computed over densely sampled cuboids and concatenated into large FV; after reducing the dimensionality of those, they compute a second (stacked) fisher vector representation for the videos. Ni et al. [18] cluster trajectories into spatio-temporal groups and learn to assign a weight to each spatio-temporal group in such a way that more discriminative (weighted) fisher vectors are obtained, attenuating the effect of irrelevant groups of trajectories for a particular action. In [8], fisher vectors are computed in a per-frame basis. The final representation for classification is not the fisher vectors in this case, but the parameters of a learned model, e.g. a linear model, that explains the evolution of the fisher vector features throughout the video.

There exist other successful approaches than those based on iDT. Ma et al. [15] explore hierarchical, spatial, and temporal relations among spatio-temporal sub-volumes, transforming the video into a graph composed of discriminative and frequent tree-like structures. An action classifier categorizes actions based on the detection responses of the trees. Miao et al. [17] take advantage of compressed video features, such as motion vectors and discrete cosine transformation coefficients to speed up to 100 times the iDT feature extraction process with comparable results. Alfaro et al. [1] use a set of pooled key-sequences to quantify relative local intra- and inter-class similarities by projecting the

key-sequences to a bank of dictionaries encoding patterns from different temporal positions or action classes. Yang et al. [33] jointly learn a high-level representation by combining a hierarchical generative model (that represents actions by distributions over latent spatial temporal patterns) and discriminative max-margin classifiers in a unified Bayesian framework. Fernando et al. [6] propose a hierarchical rank pooling – based on [8] –, but on video segments; the first layer performs rank pooling on CNN feature maps and subsequent layers on the result of previous rank pooling operations. [9] integrates the rank pooling process in end-to-end learning with CNNs.

In the context of CNNs and deep-learning, we find an increasing number of works performing action recognition. However, several major problems must be addressed when performing action recognition with such models; one of them being the size of the input layer. Researchers often feed the nets with clips of only few frames (instead of entire videos) in order to avoid dealing with much bigger models [13]. Then, the obtained prediction scores over those sampled clips are averaged in order to obtain a prediction for the whole video. Having small clips of fixed duration solves the problem of having variable-sized inputs. However, clips do not provide information for learning long-term spatio-temporal relations. For this, others use the outputs of CNNs as input to temporal models such as recurrent neural networks or long-short term memory nets [5, 24]. On the other hand, it is difficult to make these models exploit motion information. To ensure this, optical flow is often pre-computed and fed along with RGB frames in a two-stream convolutional networks [25]. This demonstrates the importance of hand-crafted methods that can boost the performance of deep learning methods. In this sense, we can find CNNs being used as effective feature extractors. Wang et al. [31] enrich the iDTs with convolutional spatial and temporal features, while de Souza et al. [4] effectively integrate iDT framework with a deep classifier, obtaining state-of-the-art results. Other authors directly build a dynamic image representation from videos [3] to take advantage of the full power of CNNs for image classification.

Our darwintree proposal reduces the high computation and memory requirements of deep-learning methods at the same time that achieves state-of-the-art results on two benchmark datasets. Our method does not require large amounts of data, neither clipping segments nor temporal resizing of the video. It naturally handles long-term temporal dependencies and exploits motion information by using specifically designed motion features (MBH) for the task of action recognition.

3. Method

The proposed system for action recognition is shown in Figure 2. For a particular video, we first extract improved dense trajectories (see Section 3.1). Then, we run a divisive hierarchical clustering algorithm based on the spectral embedding using Nyström method on a matrix of tracklet similarities (see Section 3.3). In a third stage, we use the derived binary tree structure to construct two main mid-level representations: one modeling the evolution of tracklet features on nodes and another modeling the tracklet features at tree branch level (see Sections 3.4 and 3.5). As base features, we use the well known Fisher Vectors (FV). In case of node-videodarwin, FV are computed per frame, whereas in the case of branch-videodarwin one FV is used as a global node descriptor. We use a kernel function able to compute the similarity of two trees based on pairwise similarities of mid-levels (Section 3.6). Finally, we apply binary SVM classifiers for the actual action recognition. Prediction scores from different kernels are fused in an early fusion approach by using a linear SVM classifier. The method is unsupervised until the classification part.

Next, we describe in detail each stage of darwintrees.

3.1. Extraction of trajectories and trajectory features

We use improved dense trajectories from [29]. The algorithm relies on the computation of dense optical flow fields from which local point trajectories are constructed by tracking a point during L frames. In the improved version, the authors used an homography to cancel out the camera motion from the optical flow.

Improved dense trajectories are characterized using both its relative spatial displacement throughout time and a set of state-of-the-art descriptors extracted on image patches along their trajectories. In particular, we use the Motion Boundary Histogram (MBH) descriptor, i.e. the first derivative of the optical flow, which demonstrated to be very effective for action recognition due to its robustness to camera translation and moving background. As in [29], a MBH histogram is extracted for each of the $n_x \times n_y \times n_t$ cells around the trajectory and altogether concatenated in a feature vector representing the trajectory. For the sake of simplicity, all parameters and details related to the trajectory extraction are kept as in the original paper.

3.2. Clustering of trajectory paths into binary tree structures

Following the approach of [11], we cluster the extracted trajectories on each particular video, each trajectory instance being represented by its spatio-temporal positions $\{\mathbf{x}, \mathbf{y}, \mathbf{t}\}$ and velocities $\{\mathbf{v}_x, \mathbf{v}_y\}$. Note each of these features is a feature vector of size L, the length of the trajectory's path. Then, clustering procedure consists of two parts, the construction of a similarity matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ among the n trajectories within the video and the spectral divisive clustering.

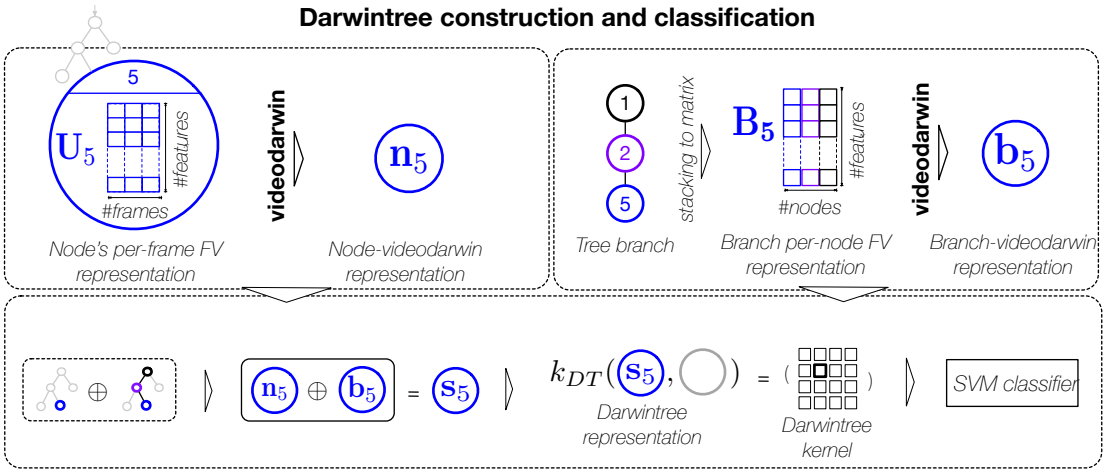

Figure 2: Proposed framework for action recognition. First, node and branch representations are created. The darwintree representation is constructed from the concatenation of the $\mathbf{n}$ and $\mathbf{b}$. Finally, the darwintree kernel is computed and input to a support vector machine (SVM) classifier.

As in [11], we first filter out the more sparse trajectories along the video given a sparsity criterion. For computing the sparsity of a trajectory in the i-th frame, we average the Euclidean distances over the set of k neighboring trajectories and within the temporal window $[i-r, i+r]$ using their mean spatio-temporal position $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{t}})$. For our implementation, we use $k = 30$ and $r = 5$ (as in [11]). Trajectory instances deviating more than one standard deviation with respect to the mean sparsity are removed. This forces around 10% of the more sparse trajectories to be removed.

Next, we compute a pairwise similarity matrix among the trajectories of the video for each feature in the set of $\mathcal{F} = \{\mathbf{x}, \mathbf{y}, \mathbf{t}, \mathbf{v}_x, \mathbf{v}_y\}$. For this purpose, we use a RBF Gaussian kernel: $k(f, f') = exp(-\gamma d(f, f'))$, with $\gamma = 1$, $d(\cdot, \cdot)$ representing the Euclidean distance among two feature vectors of size L, and $f \in \mathcal{F}$. Finally, similarity matrices of different features are aggregated into matrix $\mathbf{W}$ by their element-wise product. $\mathbf{W}$ can be seen as the adjacency matrix of a graph weighting pairwise affinities of trajectories from which we want to perform optimal recursive bi-partitioning cuts in order to eventually construct our binary tree-form structures.

Given a pairwise affinity matrix such as $\mathbf{W}$, we can use spectral grouping/clustering, that is, to embed the trajectories into a projection in the eigenvector space from which we can compute the actual clusters. However, having on the order of $n = 10^6$ trajectories makes the computation of $\mathbf{W}$ hard for any eigensolver. Nyström approximation method [10] instead, allows to use a small portion of the trajectories to extrapolate the results and obtain the approximate leading eigenvectors we need. After that, we use the divisive hierarchical clustering algorithm proposed by [11] to recursively threshold on the leading eigenvectors' values and build the corresponding unordered binary tree. As in [11], we use two hyper-parameters that define the maximum ($M = 2,000$) and minimum ($m = 200$) number of trajectories per node that help keeping the trees balanced.

3.3. Tree-based mid-level representations

At this point, we have obtained a tree-form representation for a video with nodes defining groups of trajectories. Since trajectories are local features, we first build a mid-level global representation for these nodes, so that we can later calculate the similarity between a pair of trees. [11] proposed a bag-of-words (BoW) as a mid-level representation for the nodes. Instead, we compute videodarwin [8] at two levels: node and branch. We also compute the more obvious but natural extension to BoW that are the *fisher vectors* [21] which indeed serve as the base for node-videodarwin computation.

3.4. Node videodarwin

The key idea behind VideoDarwin is to model how the features evolve throughout time. It has demonstrated a superior performance than other representations such as fisher vectors for action recognition tasks [8]. Its main limitation however is the modeling of large sequences, that might suffer from the temporal smoothing problem. The spatio-temporal decomposition from Section 3.3 provides relatively shorter video segments that are less likely to suffer from this problem.

In order to compute the VideoDarwin representation of a node, we first extract its per-frame FV representation. Each node has a temporal extent depending on the time interval spanned by the group of trajectories, $[F_{\text{begin}}, F_{\text{end}}]$. Let us denote then $\mathbf{U} \in \mathbb{R}^{2DK \times F}$ as a node's FV representation over time, where $F = F_{\text{end}} - F_{\text{begin}} + 1$ is number of frames spanned by the set of trajectories associated to that partic-

ular node, D is the dimensionality of the MBH descriptor, and K the number of Gaussian components used to estimate the FV's GMMs.

Next step is to smooth the variations of the features over time. This proved to give more stable results. For this purpose, the authors of [8] provided different solutions, the one achieving best results being a time varying mean. Thus, we define the smoothed U as $\mathbf{V} \in \mathbb{R}^{2DK \times F}$, its columns calculated as:

$$\mathbf{m}_i = \frac{1}{i} \sum_{\tau=1}^{i} \mathbf{U}_{:,\tau} \ , \tag{1}$$

$$\mathbf{V}_{:,i} = \frac{\mathbf{m}_i}{||\mathbf{m}_i||_1} \ , \forall i \in \{1 \dots F\} \ . \tag{2}$$

where the notation ":" indicates "all elements" in the corresponding matrix dimension. Therefore, $\mathbf{U}_{:,\tau} \in \mathbb{R}^{2DK \times 1}$ is simply the τ-th column vector of $\mathbf{U}$.

Given $\mathbf{V}$, there exist several possibilities for computing VideoDarwin, either using a linear *Support Vector Regressor* (SVR) or *Rank Support Vector Machine* (RankSVM). We stick to linear SVR, since it provides equally good results at lower computational cost – as stated in [8]. Hence, let us denote $\nu(\cdot, \cdot)$ a function with the data and the desired outputs as parameters returning the learnt model parameters $\mathbf{w}$, i.e., the videodarwin representation. Then:

$$\mathbf{w} = \mathbf{w}^+ \oplus \mathbf{w}^- = \nu(\mathbf{V}^+, (1, \dots, F)) \oplus \nu(\mathbf{V}^-, (1, \dots, F)) \ , \tag{3}$$

where $\oplus$ is the concatenation operator, $\mathbf{w}^+, \mathbf{w}^- \in \mathbb{R}^{2DK}$ are forward and reverse videodarwin representations obtained respectively from forward and reversed (columnwise) time varying means $\mathbf{V}^+$ and $\mathbf{V}^-$ from $\mathbf{U}$. The calculation of $\mathbf{V}^-$ is done by re-defining $\mathbf{m}_i$ such that it is calculated backwards: $\mathbf{m}_i^- = \frac{1}{(F-i)} \sum_{\tau=1}^{i} \mathbf{U}_{:,(F-\tau)}$. The reversed representation helps to compensate for the fact that the forward accumulation of $\mathbf{m}_i^+ \triangleq \mathbf{m}_i$ does not take into account future information, i.e. $\{\mathbf{U}_{:,\tau} \,|\, \tau > i\}$.

3.5. Branch videodarwin

Even though VideoDarwin was initially developed to model changes in the temporal dimension, we propose its use in tree branches. We define a branch as the path from a particular node up to the root. This way, in case of having global representations for each node – instead of per-frame – we can model evolution of node features at branch level. That is, what features become relevant when isolating a group of trajectories. Or what features grow in importance when we keep merging going up until the root node is reached.

For our purpose, we compute a global FV representation $\mathbf{u}$ for each node that will serve as a base representation prior to applying videodarwin. We then define a branch matrix representation for the i-th node as:

$$\mathbf{B}_i = \mathbf{u}_i \oplus \mathbf{u}_{\lfloor i/2^1 \rfloor} \oplus \mathbf{u}_{\lfloor i/2^2 \rfloor} \oplus \dots \oplus \mathbf{u}_1 \ , \tag{4}$$

where $\lfloor \cdot \rfloor$ refers to the floor (rounding down) operation.

Analogously to $\mathbf{U}_i$ from Section 3.4, we perform videodarwin on $\mathbf{B}_i$. The difference is that, in this case, the obtained representation explains the changes of $\mathbf{u}$ features when either ascending from the i-th node to the root (forward videodarwin) or descending from the root to the i-th node (reverse videodarwin).

It is not possible to construct a path solely from the root node. Therefore, we come up with as many branches as intermediate nodes and leafs. Once computed the branch representations, we define a tree structure: $\mathcal{T} = \{\mathbf{r}, \mathcal{N}, \mathcal{B}\}$, where $r \triangleq \mathbf{w}_1$ is the root node representation, $\mathcal{N} = \{\mathbf{n}_i \,|\, i > 1\}$, and $\mathcal{B} = \{\mathbf{b}_i \,|\, i > 1\}$.

3.6. Darwintree classification

In order to perform classification, we transform it to a more compact representation than unordered binary tree structures with a variable number of nodes and branches – one that can be input in any state-of-the-art discriminative classifier. The authors of [11] have already proven the accumulation of pair-wise node similarities to be effective for tree discrimination with their *All Tree Edge Pairs* (ATEP) kernel. They reported better results by using edge representations, i.e. the concatenation of child and parent representations, than by solely using the child representation. In our work, we define our own representation by combining node-branch representations for the computation of a darwin-tree kernel. Let us define this joint node-branch representation, named darwintree representation, as the concatenation of node- and branch-videodarwin, that is: $\mathbf{s} = \mathbf{n} \oplus \mathbf{b}$. Then, we compute the darwintree kernel k_{DT} based on the pair-wise similarity of darwintree representations:

$$k_{DT}(\mathcal{S}_i, \mathcal{S}_j) = \frac{1}{|\mathcal{S}_i||\mathcal{S}_j|} \sum_{\mathbf{s}_i \in \mathcal{S}_i} \sum_{\mathbf{s}_j \in \mathcal{S}_j} \phi(\mathbf{s}_i, \mathbf{s}_j) \ , \ \forall i, j > 1 \tag{5}$$

where $\phi(\cdot, \cdot)$ can be any valid kernel function, e.g. dot product for a linear mapping.

Note $\mathbf{s}_1$ is omitted in this case because there is no possible branch construction from a node (root) to itself. Once the darwin kernel is calculated, we can perform action recognition.

4. Experiments

Before presenting the results, first we describe the considered datasets and evaluation protocol, and method details.

Datasets We tested our approach on UCF sports actions dataset [23], Highfive [20], and Olympic Sports [19]. UCF sports actions contains 150 examples and 10 classes of actions from different sports, presenting different backgrounds and camera movement. In our experiments, we

used the standard 103/47 train/test split. Highfive consists of 300 examples of human interactions from TV shows, from which 200 are handshake, high five, hug, kiss, whereas the 100 remaining ones are negative examples. For validation, we followed the 2-fold cross validation provided along with the dataset. We report both accuracy and mean average precision (mAP). Finally, Olympic Sports contains 783 instances from 16 classes partitioned in a 640/143 train/test holdout split. The results obtained are expressed in mAP.

Code implementation We constructed the unordered binary trees of trajectories using public code[1] (with default parameters) provided by the authors of [11]. On the other hand, we used the videodarwin implementation[2] from [8] as a base for the construction of our darwintrees. For classification and evaluation metrics, we used Python's *sklearn* machine learning library. In particular, for multi-class classification we choose sklearn's one-vs-rest classification over the one-vs-one LibSVM's implementation. Moreover, Eq. 5 is optimized for GPU computation with Python's *pycuda* library.

Trajectory features, GMMs, fisher vectors, and spectral clustering We extracted MBH features along the trajectories, applied the "square-root trick" as in [30], and reduced their 192-dimensional descriptors by a factor of 0.5 using PCA. We used 256 mixtures on our GMMs and 1 million MBH descriptors for their training. This yielded fisher vectors of dimensionality $2 \cdot 96 \cdot 256 = 49,152$. As suggested by [7], prior to the videodarwin computation we applied posneg mapping first to the fisher vectors; this is $\mathbf{v} = \sqrt{\mathbf{v}_{\text{pos}} \oplus \mathbf{v}_{\text{neg}}}$, where $\mathbf{v}_{\text{pos}}$ is the $\mathbf{v}$ with all zeros except for the positive coefficients and $\mathbf{v}_{\text{neg}}$ all zeros with all the negatives turned into positive. After posneg mapping, we also applied l2-normalization. For the spectral clustering, we stick to the parameters given by [11], except for the maximum number of tree levels (default value is 62). We experimentally found that in very deep trees, deeper nodes tend to be noisy and cause great impact in the performance of node-videodarwin. We experimentally found a value of 4 levels to be a conservative value. See the results for UCF Sports Actions in Figure 3.

VideoDarwin, kernel maps, and classification Since videodarwin representations consist of both forward and reverse videodarwin (depending on the direction of the mean time varying operation) parts, we come up with a final representation that doubles the size of the fisher vectors, i.e., 98,304 dimensions. This gives a descriptor of $N \times 98,304$ per video of N frames. For classification, we kernel mapped the VideoDarwin representation using "RootSIFT" [2] and l2-normalized them. As a last step, different mid-level representations were fused at kernel level and the weights assigned were cross-validated. A normalization factor is applied to the kernels before the aggregation, consisting of dividing each kernel by the maximum value of the diagonal. This is because otherwise when comparing a tree to itself the similarity is not 1. For all our experiments, we fixed the C parameter of the SVM classifiers to 100.

[1]Tree structure and hierarchical divisive algorithm for spectral clustering: `gist.github.com/daien`.

[2]Videodarwin code: `bitbucket.org/bfernando/videodarwin`.

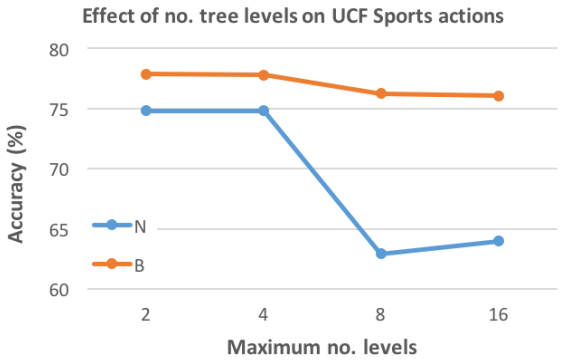

Figure 3: Performance varying the number of maximum tree levels on UCF Sports actions in terms of accuracy (%). Experiments in the validation dataset showed videodarwin on noisy deeper tree nodes causes the our node representation (N) to underperform in comparison to the branch representation (B).

Method	UCF [23] (ACC)	Highfive [20] (mAP)		
		Fold 1	Fold 2	TOTAL
Node-VD	85.11	76.55	70.41	73.48
Branch-VD	80.85	76.25	**72.53**	74.39
Darwintree (DT)	**91.49**	76.04	70.37	73.21
VD+DT	**91.49**	**79.24**	72.32	**75.78**

Table 1: Results of the different methods in the two benchmarking datasets for node-videodarwin, branch-videodarwin, darwintree (DT), and the combination of DT with holistic videodarwin (VD).

Action classification (quantitative) results We illustrate our results in the benchmarking datasets on Table 1, in which we compare our different approaches among them: node-videodarwin (N), branch-videodarwin (B), the combination of both (Darwintree or DT), and the combination of the latter with the holistic representation (VD+DT). Despite DT and VD+DT got the same results, we found VD+DT to be potentially better from training data: +2.81% (79.80 against 76.99) on average. We also compared our approach (VD+DT) to the holistic videodarwin representation in UCF Sports Actions and obtained better performance: 91.49% vs 87.23%.

To provide more insight about the classification accu-

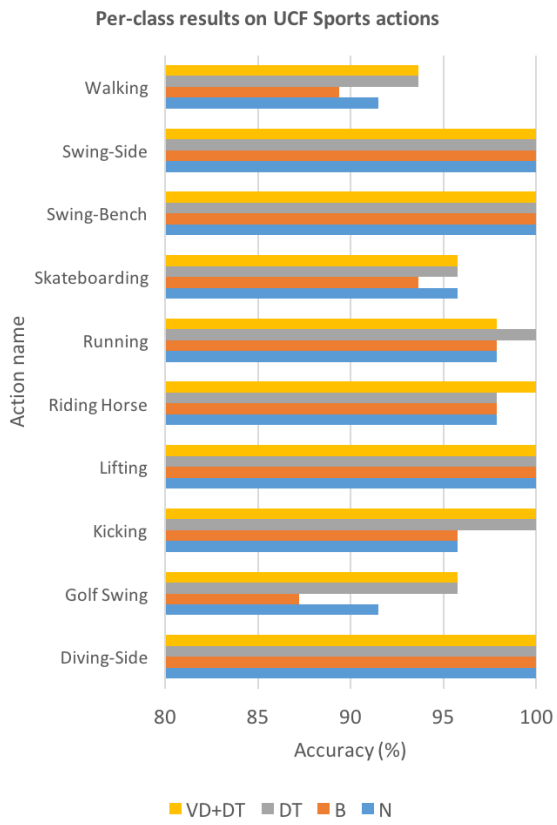

Figure 4: Results for the different action classes of UCF Sports actions in terms of accuracy (%).

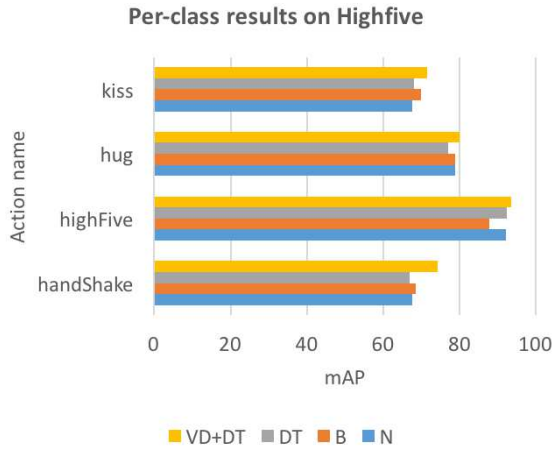

Figure 5: Results for the different action classes in terms of mean average precision (mAP).

racy, we show the results for the different action classes for the UCF dataset in Figure 4.

Since UCF and H5 are fairly small datasets, we further validated our method in Olympic Sports dataset. It consists of 783 action instances and 16 action classes. Those classes are: "basketball layup", "bowling", "clean and jerk", "discus throw", "diving platform 10m", "diving springboard 3m", "hammer throw", "high jump", "javelin throw", "long jump", "pole vault", "shot put", "snatch", "tennis serve", "triple jump", and "vault". For the validation, we use the standard 640/143 train/test split. In our experiment, we compared H, N, B, NB, and H+NB representations in Table 2. Despite N, B, and the combination NB obtained poorer results than H, combining NB with H yields slight improvement of +0.5% mAP points with respect to H.

Method	mAP
Videodarwin (H)	88.34
Node-vd (N)	83.17
Branch-vd (B)	87.70
NB	84.38
H+NB	**88.84**

Table 2: Olympic Sports dataset [19].

Qualitative results These consist on the visualization of sequence frames with overlayed trajectory clusters, along with the predicted categories by our proposed method. For the sake of simplicity, hereinafter, we refer to the holistic videodarwin as "H", node-videodarwin as "N", branch-videodarwin as "B", the early fusion of "N" and "B" as "NB", and the combination "NB" with "H" at kernel-level as "H+NB".

In Figure 6, we illustrate some results on UCF Sports Actions [23]. Several sequences are shown in series of 3 frames evenly spaced time with the extracted trajectories' clusters on top. Notice the compactness of the clusters in both space and time and coherence. Moreover, similar actions and viewpoints have similar cluster decompositions (see the two "Golf-Swing-Back" action examples in Figure 6a and Figure 6b). In simpler actions, as Figure 6d and Figure 6e, the number of trajectories is smaller and, hence, decompositions are simpler. While in Figure 6d, the one and only decomposition is throughout the temporal dimension (first frames being yellow cluster and latter ones the blue cluster), in Figure 6e is along the spatial dimension (upper body being yellow and lower body the blue cluster).

Once can see that, except for 6d example, our method H+NB is able to predict correctly the actual groundtruth class (GT), while H was wrong in most cases (only 2/5 hits).

Note also that in all except for 6d, B predicted the wrong class. However, for all of those, NB was able to predict it right despite N also being wrong, as it was the case for 6b and 6d. This proves NB learns to model the complementary information provided by N and B.

NB tends to correct bad results gotten by H, as seen in 6b or 6e. Nonetheless, it is also possible that the H representation causes a bad final prediction, as in 6d. In general, however, H+NB proved to be more effective for the classification task than only NB.

(a) **Golf-Swing-Back 002**. GT=1, H=1, N=1, B=9, NB=1, H+NB=1.

(b) **Golf-Swing-Back 005**. GT=1, H=2, N=2, B=9, NB=1, H+NB=1.

(c) **Kicking-Front 010**. GT=2, H=2, N=2, B=5, NB=2, H+NB=2.

(d) **Run-Side 001**. GT=5, H=9, N=9, B=5, NB=5, H+NB=9.

(e) **Skate-Boarding-Front 004**. GT=6, H=1, N=6, B=1, NB=6, H+NB=6.

Figure 6: Trajectory clusters on 3 frames evenly spaced in time for 3 different UCF Sports Actions' examples [23]. See in the captions of (a)-(e) of subfigures the groundtruth label (GT) and the output of our different methods (H, N, B, NB, and H+NB). Classes are "Diving-Side" (1), "Golf Swing" (2), "Kicking" (3), "Lifting" (4), "Riding Horse" (5), "Running" (6), "Skateboarding" (7), "Swing-Bench" (8), "Swing-Side" (9), and "Walking" (10).

Comparison to state-of-the-art methods In Table 3 and Table 4, we illustrate state-of-the-art results compared to ours. As shown, our method achieves state-of-the-art results in both UCF Sports Actions and Highfive benchmarking datasets using the standard metrics and evaluation protocols, improve the results in +0.7% and +6.4 points respectively.

Discussion Our results demonstrate the effectiveness of combining node and branch videodarwin representations, that is, the final darwintree representation. In Highfive, the holistic representation pushed further the mAP performance obtained by our method. Our method does not need lots of training data in order to generalize. Moreover, the divisive clustering technique based on spectral embedding is an unsupervised technique that does not require annotated training data. Interestingly, the proposed pipeline is general enough to be applied to any kind of video sequence classification problem. Once a representation per time instant is build, the method can be directly applied.

Method	Accuracy (%)
Ours (VD+DT)	**91.5**
Karaman et al. [12](2014)	90.8
Ma et al. [15](2015)	89.4
Wang et al. [32](2013)	85.2
Ma et al. [16](2013)	81.7
Raptis et al. [22](2012)	79.3

Table 3: UCF-sports dataset [23].

Method	mAP
Ours (VD+DT)	**75.8**
Wang et al. [28](2015)	69.4
Karaman et al. [12](2014)	65.4
Ma et al. [15](2015)	64.4
Gaidon et al. [11](2014)	62.4
Ma et al. [16](2013)	36.9
Patron-Pérez et al. [20](2012)	42.4

Table 4: Highfive dataset [20].

5. Conclusion and future work

We proposed a novel representation for action/activity recognition on RGB videos. We modeled the evolution of iDT features on groupings of trajectories. The groupings were obtained by using a recursive clustering algorithm that performed a hierarchical decomposition of the video's cloud of trajectories. Then, we modeled the evolution of features throughout both the frames of a subpart and descending/ascending paths in a branch of the tree. For the final classification, we constructed a kernel representation combining the two proposed representations. Moreover, our method shows further improvement when used together with holistic videodarwin. We achieved better results than current state-of-the-art on two benchmark datasets (UCF Sports Actions and Highfive) for action recognition.

The pipeline is applicable to any pattern recognition problem once a rich representation is obtained at a given time instant. As future work, we propose to integrate CNN features as a new source of information and explore the use of darwintrees for spatio-temporal localization of actions.

6. Acknowledgments

This work has been partially supported by the Spanish project TIN2016-74946-P (MINECO/FEDER, UE) and CERCA Programme / Generalitat de Catalunya.

References

[1] A. Alfaro, D. Mery, and A. Soto. Action recognition in video using sparse coding and relative features. In *Proceed-*

ings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2688–2697, 2016.

[2] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2911–2918. IEEE, 2012.

[3] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition CVPR*, 2016.

[4] C. R. de Souza, A. Gaidon, E. Vig, and A. M. López. Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition. In *European Conference on Computer Vision*, pages 697–716. Springer International Publishing, 2016.

[5] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[6] B. Fernando, P. Anderson, M. Hutter, and S. Gould. Discriminative hierarchical rank pooling for activity recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. 2016.

[8] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5378–5387, 2015.

[9] B. Fernando and S. Gould. Learning end-to-end video classification with rank-pooling. In *International Conference on Machine Learning*, pages 1187–1196, 2016.

[10] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):214–225, 2004.

[11] A. Gaidon, Z. Harchaoui, and C. Schmid. Activity representation with motion hierarchies. *International Journal of Computer Vision*, 107(3):219–238, 2014.

[12] S. Karaman, L. Seidenari, S. Ma, A. Del Bimbo, and S. Sclaroff. Adaptive structured pooling for action recognition.

[13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[14] G. Lev, G. Sadeh, B. Klein, and L. Wolf. Rnn fisher vectors for action recognition and image annotation. In *European Conference on Computer Vision*, pages 833–850. Springer, 2016.

[15] S. Ma, L. Sigal, and S. Sclaroff. Space-time tree ensemble for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5024–5032, 2015.

[16] S. Ma, J. Zhang, N. Ikizler-Cinbis, and S. Sclaroff. Action recognition and localization by hierarchical space-time segments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2744–2751, 2013.

[17] J. Miao, X. Xu, R. Mathew, and H. Huang. Residue boundary histograms for action recognition in the compressed domain. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2825–2829, Sept 2015.

[18] B. Ni, P. Moulin, X. Yang, and S. Yan. Motion part regularization: Improving action recognition via trajectory selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3698–3706, 2015.

[19] J. C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European conference on computer vision*, pages 392–405. Springer, 2010.

[20] A. Patron-Perez, M. Marszalek, I. Reid, and A. Zisserman. Structured learning of human interactions in tv shows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2441–2453, 2012.

[21] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *European Conference on Computer Vision*, pages 581–595. Springer, 2014.

[22] M. Raptis, I. Kokkinos, and S. Soatto. Discovering discriminative action parts from mid-level video representations. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1242–1249. IEEE, 2012.

[23] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[24] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584. IEEE, 2015.

[25] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[26] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *arXiv preprint arXiv:1604.04494*, 2016.

[27] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.

[28] H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *International Journal of Computer Vision*, pages 1–20, 2015.

[29] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3551–3558, 2013.

[30] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *ICCV 2013 - IEEE International Conference on Computer Vision*, pages 3551–3558, Sydney, Australia, Dec. 2013. IEEE.

[31] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4305–4314, 2015.

[32] L. Wang and H. Sahbi. Directed acyclic graph kernels for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3168–3175, 2013.

[33] S. Yang, C. Yuan, B. Wu, W. Hu, and F. Wang. Multi-feature max-margin hierarchical bayesian model for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.